Spatial Modeling and Classification of Corneal Shape

Keith Marsolo*, Michael Twa†, Srinivasan Parthasarathy*, and Mark Bullimore†

*Department of Computer Science and Engineering
Email:{marsolo, srini}@cse.ohio-state.edu
†College of Optometry
Email:{MTwa, MBullimore}@optometry.osu.edu
*,†The Ohio State University, Columbus, OH

Abstract

One of the most promising applications of data mining is its use on biomedical instrument data to assist in patient diagnosis. Any method of biomedical data analysis intended to support the clinical decision-making process should meet several criteria: it should capture clinically relevant features, should be computationally feasible and should provide easily interpretable results. In an initial study, we examined the feasibility of using a spatial transformation in conjunction with a decision tree classifier to distinguish between diseased and non-diseased eyes using biomedical instrument data. Here we provide a comprehensive follow-up to that work, addressing several of its limitations while presenting a number of meaningful extensions. First, we examine an additional spatial transformation to determine whether it provides any increase in classification accuracy. Second, we have designed and implemented a novel meta-classifier that provides superior results when compared to traditional classifiers. We present our results comparing classification accuracy as it relates to dataset and transformation resolution over a more challenging, multi-class dataset. Third, we provide a novel technique for the visualization of decision trees that can assist medical experts during patient diagnosis. Finally, we present several optimizations to our algorithm that are designed to aid in the high-performance deployment of our classification system.

1 Introduction

The use of biomedical diagnostic systems to support clinical decision-making presents several challenges that are well-suited to knowledge discovery and data mining methods. In general, this problem consists of extracting useful patterns of information from large quantities of data where features often have complex interactions and fail to conform to the assumptions of traditional statistical modeling methods. Data obtained from diagnostic machines are inherently multidimensional, often noisy, and generally specific to the instrument producing it. Therefore, summarizing such data in simple terms without losing potentially useful information becomes an extremely difficult task. A natural conflict also exists between the need to simplify the data to make it more interpretable and the associated risk of sacrificing information relevant to decision support.

In an earlier study, we reported promising results on a problem of classifying clinical eye disease [18]. Specifically, we proposed a procedure whereby one can model the surface features of the outer eye using Zernike polynomials [19] and subsequently use the coefficients produced by the polynomial transformation to build a decision tree classifier that can discriminate between diseased and non-diseased eyes. Here we present an in-depth follow-up with four significant extensions that greatly expand the scope of our previous work as well as address some of its limitations. Before proceeding, we should note that while this study deals primarily with the classification of eye disease, it has implications that can be applied to any problem that deals with the model-based mining of data. Our extensions and contributions are described below.

First, we evaluate the use of *alternate spatial models* to represent the data. We examine the use of pseudo-Zernike polynomials, a related, but less common, orthogonal polynomial series that has been shown to be more tolerant to noise effects than Zernike polynomials [5]. We also examine the use of wavelets, which are widely used in many biomedical imaging applications [11]. We compute the transformations on several different resolutions of our dataset and examine the impact of changing the polynomial order as it relates to computation time and model error.

Second, we present a *novel meta-learning strategy*

that combines classifiers built on different spatial models. We also demonstrate that for our application, it can increase classification accuracy and can be combined with other meta-learning techniques such as bagging [3] and boosting [8] to further improve classification results. We compare our strategy against several different classifiers: C4.5 [13], Voted Feature Intervals [7], Naïve Bayes [10], and a neural network [1]. We run our classification experiments on more challenging dataset when compared to the data used in our initial study [18]. Complicating the issue here is the presence of eyes that have undergone laser refractive surgery.

Third, we have designed and developed a novel method of visualizing our decision tree results. We create a multi-dimensional surface generated from the values of the polynomial coefficients that correspond to the splitting nodes of the decision tree. We have shown our results to practicing clinicians and preliminary validation from them indicates there is an anatomic correspondence between this surface and the corneal shape of patients with degenerative eye disease.

Finally, we have implemented a number of high-performance versions of our transformation algorithm that are designed to take advantage of the benefits of distributed computing. These implementations are specifically tailored for a health center environment without access to a super-computer. Designed to utilize a workstation's spare computing cycles, it is possible to achieve high-performance results without a high-performance infrastructure. We discuss the motivation behind our optimizations and provide computation times for each.

2 Clinical Application

Sensing a visual image begins when light entering the eye is focused by the cornea, passes through the pupil, and is further focused by the lens forming an image on the retina located at the back wall of the eye. Since light must pass through the cornea, clear vision depends very much on the optical quality of the corneal surface, which is responsible for nearly 75% of the total optical power of the eye. Normal corneas tend to have an aspheric profile that is more steeply curved in the center relative to the periphery where the cornea joins the white tissue known as the sclera. Subtle distortions in the shape of the cornea can have a dramatic effect on vision. For this reason, the shape of the corneal surface is measured clinically for the treatment and diagnosis of eye disease. The process of mapping the surface features on the cornea is known as corneal topography. The most common method of determining corneal shape is to photograph the reflection of a series of concentric rings imaged by the corneal surface. Any distortion in corneal shape will cause a distortion of the concentric rings. By comparing the size and shape of the photographed rings with their known dimensions, it is possible to mathematically derive the topography of the corneal surface. Interpretation of this topography is the issue we addressed in this research. Many methods of interpretation exist and in clinical practice, domain expertise is the usual basis for interpretation. There is a lack of broadly accepted standards among domain experts and instrument manufacturers, which makes interpretation largely a qualitative task of pattern recognition.

The use of corneal topography has rapidly increased in recent years because of the popularity of refractive surgery and the decreased cost of powerful personal computers. Refractive surgery is an elective surgical treatment intended to reduce one's dependence on glasses or contact lenses. The most common surgical treatment performed for this purpose is laser assisted in-situ keratomileusis (LASIK). Corneal topography is used to screen patients for corneal disease prior to this surgery and to monitor the effects of treatment after surgery.

Another clinical application of corneal topography is the diagnosis and management of keratoconus. Keratoconus, a progressive, non-inflammatory corneal disease, distorts corneal shape and results in poor vision that cannot be corrected with ordinary glasses or contact lenses. Patients with keratoconus frequently seek refractive surgery due to their poor vision. However, such treatment exacerbates their corneal distortion and frequently leads to corneal transplant [20]. Thus, corneal topography is a valuable tool for diagnosis and management of keratoconus as well as for the prevention of inappropriate refractive surgery in this patient group.

The dataset used in this experiment consisted of examinations of 254 eyes obtained from a clinical corneal topography system, the Optikon Keratron[1]. We studied three patient groups that represent a wide variety of corneal shapes (number of patients listed in parentheses): 1) normal (n=119), 2) distorted, but generally flatter than normal curvature (post-operative LASIK, n=36), and 3) distorted and generally steeper than normal curvature (keratoconous, n=99). The data files from the corneal topographer consist of a three-dimensional matrix of approximately 7000 spatial coordinates arrayed in a polar grid. The height of each point z is specified by the relation $z = f(\rho, \theta)$, where the height relative to the corneal apex is a function of radial distance from the origin (ρ) and the counter-clockwise angular deviation from the horizontal merid-

[1] http://www.optikon.com/products/keratron/

ian (θ). Each patient record consists of two data files: one containing the values of the height data z and the other containing radial distance ρ. Each concentric ring consists of 256 data points taken at a known angle θ from the origin. In the next section we describe the methods developed to transform this spatial data and classify corneal shape.

3 Spatial Representations

In order to run our classification experiments, we need to transform the data. This is necessary for two reasons. First, we want to preserve the important features that may exist in the data and any correlations among them. Second, each data record consisted of thousands of data points. Using each point as a feature results in terrible classification performance and moreover is not scalable. Below we describe the methods and rationale for each transformation technique evaluated.

3.1 Zernike Polynomials

Zernike polynomials are a family of circular polynomial functions that are orthogonal in x and y over a normalized unit circle and consist of three elements [17]. The first element is a normalization coefficient. The second element is a radial polynomial component and the third, a sinusoidal angular component. The general form for Zernike polynomials is given by:

$$Z_n^{\pm m}(\rho,\theta) = \begin{cases} \sqrt{2(n+1)}R_n^m(\rho)\cos(m\theta) & \text{for } m > 0 \\ \sqrt{2(n+1)}R_n^m(\rho)\sin(|m|\theta) & \text{for } m < 0 \\ \sqrt{(n+1)}R_n^m(\rho) & \text{for } m = 0 \end{cases} \tag{3.1}$$

where n is the radial polynomial order and m represents azimuthal frequency. The normalization coefficient is given by the square root term preceding the radial and azimuthal components. The radial component of the Zernike polynomial, the second portion of the general formula, is defined as:

$$R_n^m(\rho) = \sum_{s=0}^{(n-|m|)/2} \frac{(-1)^s(\mathrm{n}-\mathrm{s})!}{\mathrm{s}!(\frac{\mathrm{n}+|\mathrm{m}|}{2}-\mathrm{s})!(\frac{\mathrm{n}-|\mathrm{m}|}{2}-\mathrm{s})!}\rho^{n-2s} \tag{3.2}$$

Polynomials that result from fitting our raw data with these functions are a collection of orthogonal circular geometric modes. The coefficients of each mode are proportional to the contribution of the mode to the overall topography of the original data. As a result, we can effectively reduce the dimensionality of the data to a subset of polynomial coefficients that describe spatial features, or a proportion of specific geometric modes present in the original data.

3.2 Pseudo-Zernike Polynomials

Pseudo-Zernike polynomials are another family of orthogonal polynomial functions [16]. Like Zernike polynomials, these functions are a series of circular modes orthogonal over x and y for a normalized unit circle. Pseudo-Zernike polynomials are additionally orthogonal in the radial dimension (ρ) over a unit circle. The general form for pseudo-Zernike polynomials is given by:

$$PZ_n^{\pm m}(\rho,\theta) = \begin{cases} \sqrt{2(n+1)}R_n^m(\rho)\cos(m\theta) & \text{for even } n, m \neq 0 \\ \sqrt{2(n+1)}R_n^m(\rho)\sin(|m|\theta) & \text{for odd } n, m \neq 0 \\ \sqrt{(n+1)}R_n^m(\rho) & \text{for } m = 0 \end{cases} \tag{3.3}$$

Similarly, the radial portion of the pseudo-Zernike polynomial is defined as:

$$R_n^m(\rho) = \sum_{s=0}^{n-|m|} \frac{(-1)^s(2\mathrm{n}+1-\mathrm{s})!}{\mathrm{s}!(\mathrm{n}-\mathrm{m}-\mathrm{s})!(\mathrm{n}+\mathrm{m}+1-\mathrm{s})!}\rho^{n-s} \tag{3.4}$$

There are a number of reasons for choosing Zernike and pseudo-Zernike polynomials. First, *several of the circular modes of these polynomials show strong correlation with natural anatomic features of the cornea.* Second, these polynomials were originally derived to describe optical aberrations and their geometric modes have a direct relation to the anatomical features and optical function of the eye [2, 19]. In addition, it has been shown that pseudo-Zernike polynomials are more robust with noisy data than Zernike polynomials [9]. Our previous study [18] showed that with Zernike polynomials, lower order transformations were able to capture the general shape of the cornea and were unaffected by noise. Higher order transformations were able to provide a better model, but were also more susceptible to noise. Thus, we study pseudo-Zernike polynomials to see if they result in a better model while still maintaining their resilience to noise. One disadvantage of using pseudo-Zernike polynomials is their greater computational cost. This drawback is offset by the fact that the pseudo-Zernike transformation algorithm can easily be parallelized.

How well the Zernike and pseudo-Zernike polynomial functions model abnormal corneal surfaces, such as those distorted by keratoconus and LASIK, is not well understood. Using patients from each of these categories, we make comparisons of model fidelity between each polynomial. We prefer a transformation that will faithfully capture all relevant spatial features in the raw data. Even so, we recognize the trade-offs between model accuracy, computational expense and interpretability. We calculate the difference in accuracy for each transformation method and determine how classification performance is related to accuracy.

3.3 Wavelets

Wavelet transforms have been used in a number of applications, ranging from signal processing, to image compression, to numerical analysis [6]. Wavelets also play a large role in the processing of biomedical instrument data obtained through techniques such as ultrasound, magnetic resonance imaging (MRI) and digital mammography [11]. We evaluated the application of a wavelet transformation on our data to obtain coefficients that could be used as input for a classifier. However, we encountered several issues that led us to abandon these efforts. One drawback to using wavelets is that there is no natural correlation between a wavelet coefficient and the anatomic features of the cornea. While a particular Zernike coefficient always represents the same geometric mode, such is not the case with wavelets. There is no way to guarantee the coefficients that are "important" on one image will be "important" on another, especially if those images come from two separate classification categories. This increases the difficulty of classification immensely.

A transformation's ability to capture the spatial relationships inherent in the data is one of its most important features. Another key feature is a transformation's ability to reduce the dimensionality of the data. In our dataset, each patient record contains approximately 7000 data points, although that number fluctuates depending on the resolution desired. We would like any transformation that we use to reduce that number to a more manageable size. A 7th order Zernike polynomial, for example, contains 36 coefficients. This results in a an almost 200-fold reduction in the dimensionality of the data. With wavelets, each order reduces the data by a factor of 4. To achieve the same average dimensionality reduction using wavelets as we would with Zernike-based polynomials, we would need to use a 4th order wavelet transformation. A 4th order Haar wavelet is equivalent to taking the average of the average of the average of the average. *A classifier built on such a transformation yields results that are essentially equivalent to those returned by a random classifier.*

4 Algorithms

In this section we detail the algorithms used in this study. A pseudo-code description of our spatial transformations is provided in Section 4.1. We present our novel classification strategy and visualization technique in Sections 4.2 and 4.3, respectively. Finally, we describe the high-performance implementation of our transformation algorithms in Section 4.4.

4.1 Spatial Representation

Figure 1(a) contains a pseudo-code description of the algorithm used to compute the Zernike and pseudo-Zernike polynomial transformations for each patient record. The algorithm takes as input the number of coefficient terms to compute (i.e. the polynomial order), *numCoefs* and the largest radius to examine, *maxRadius*.

The first two steps of the algorithm involve reading the radius and elevation values from the patient record and converting them from mm to μm. Once this step is complete, we are left with an array of values representing the radius and an array of values representing the elevation. Each concentric ring contains 256 points, so we know the angle for every radius value (determined by its index in the data file) and can compute an array of θ values that correspond to the angle of each data point.

The next step in the algorithm is to determine the number of data points that we must consider. This is done by finding the first data value with a radius that exceeds the user-specified parameter *maxRadius*. Since this point can come anywhere along a ring, we perform an integer division on that value and then multiply it by 256 to ensure that we include a full ring of points. We then consider all the data points in the patient record that come before this value, which we denote *numDP*.

Once we have determined the number of data points to examine, we perform the Zernike and pseudo-Zernike transformation (Figures 1(c) and 1(b), respectively) on each point. In Figures 1(b) and 1(c), n is the radial polynomial order and m represents azimuthal frequency. The values of m and n are determined based on the number of coefficients, i. The transformation process, shown in step 5 of Figure 1(a) results in a matrix of size (*numDP* x *numCoefs*). After computing the given transformation on each data point, we compute a least-squares fit of our computed z values against the z values from the original patient record. Computing the least-squares fit, shown in step 6, results in a single column matrix of size $|numCoefs|$. Finally, we compute the residual model error as a function of the polynomial order by performing a root mean squared (RMS) error calculation (steps 7-9).

4.2 Model Averaging Meta-Classification

Here we present our meta-classification strategy, which is based on the idea of averaging the results from classifiers built on multiple spatial representations to reach a classification decision. We refer to this strategy as MAM-classification (Model Averaging Meta-classification). MAM-classification is similar in

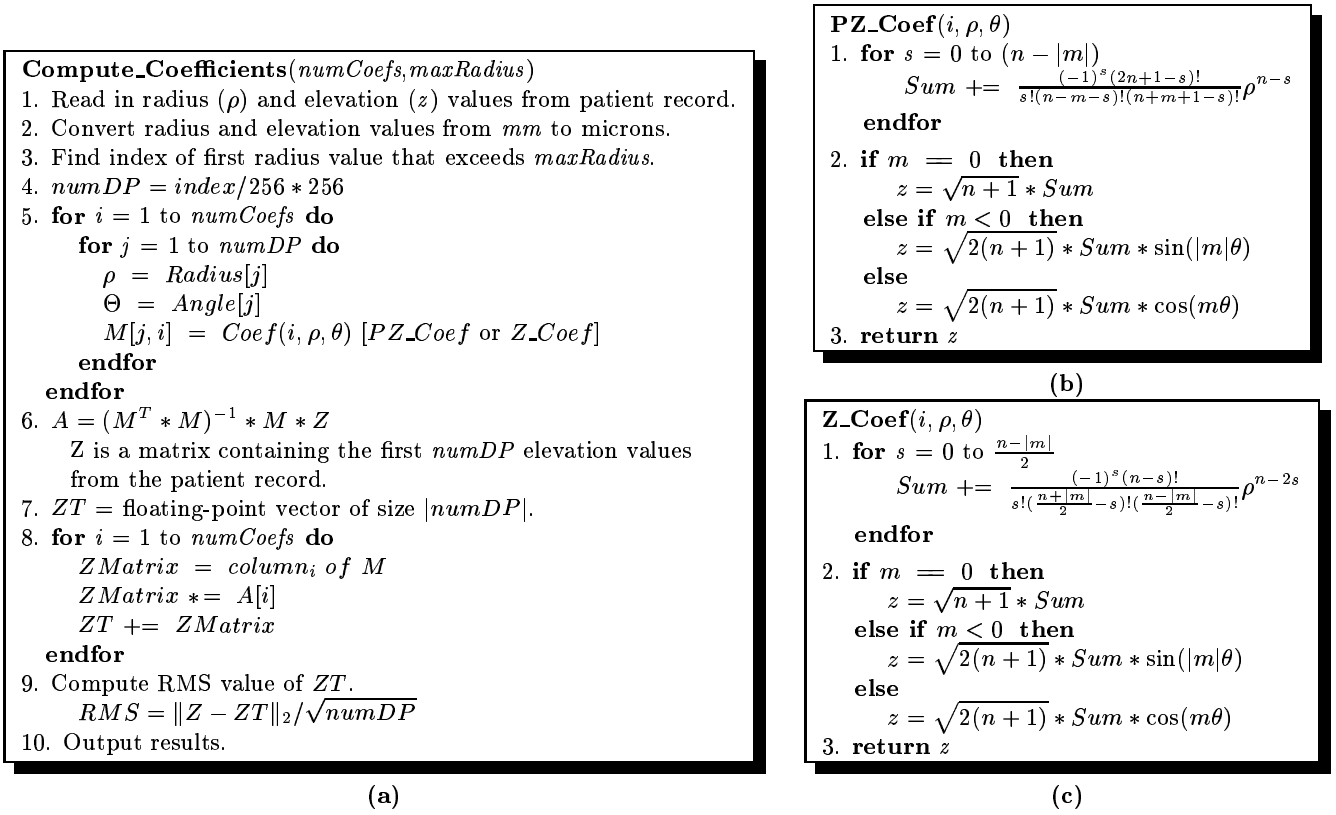

Figure 1: Pseudo-code descriptions of (a) overall algorithm, (b) pseudo-Zernike transformation and (c) Zernike transformation

spirit to boosting [8] and bagging [3]. Unlike bagging, which uses random sub-samples of the data to improve classification accuracy, we use the results of multiple spatial models. It was designed to smooth out perturbations due to *noise effects*. Intuitively, as stated in Section 3, the lower order transformations result in a model that is less detailed than those generated by the higher order transformations. However, the higher order transformations also contain noise that is not present in the models derived from the lower order transformations. *By combining the results of multiple classifiers, we hope to minimize noise effects while still retaining the benefits of the more detailed models.*

The MAM-classification algorithm is as follows: First, each individual record is converted into an input appropriate for the underlying model classifier. This step is repeated on each record for each classifier that is to be aggregated by the MAM-classifier. Second, a vote for a particular classification label is determined for each record by running the converted input through every model classifier. Finally, the MAM-classifier makes a classification decision by taking the modal label of the votes returned by the model classifiers. Thus, to create a MAM-C4.5 classifier, we would create several different C4.5 decision trees (one for each polynomial order and transformation we want to evaluate), convert each record into its corresponding Zernike and pseudo-Zernike representation, run the transformed records through their respective trees and take a modal vote of the decisions. In a similar fashion, one could construct a MAM-classifier to only aggregate the votes of the Zernike decision trees, creating a *MAM-Z* classifier.

We believe this technique may be useful for any classification problem where the features used in classification are derived using a model-based approach. For example, this idea can be used in temporal contexts if the features being used are derived from an FFT transformation.

4.3 Visualization

While accuracy is the most important factor in deciding which classification strategy to employ, another attribute that should not be ignored is the interpretability of the classification results. Decision trees have often been preferred to more accurate "black-box" classifiers because they provide more interpretable results. In medical image interpretation, decision support for a domain expert is often preferred to auto-

mated decision making by an expert system. While it is important for a system to provide a decision, it is often equally important for clinicians to know the basis for an assigned classification. Decision trees provide a more transparent view of how these classifications are derived and what features are responsible for an individual assignment.

Since the polynomial coefficients used as classification attributes in these experiments actually represent the proportional contributions of specific geometric modes, we are able to use this information to create a geometric spatial representation of the surface that lies at the boundary between classification categories. We are thereby able to use the decision tree output to provide a visualization of these results that provides more information than a simple yes or no class assignment. The specific Zernike polynomial modes that were selected as decision tree attributes are, by definition, the anatomical features that help discriminate between patient categories. *We use these polynomial coefficients in their associated splitting values to construct a multidimensional surface that aids in the discrimination between patient categories.* The creation of this surface is possible because Zernike and pseudo-Zernike polynomials are orthogonal over the unit circle in the x and y dimensions.

4.4 High-Performance Implementations

After running our initial experiments, it quickly became clear we would need a faster, more efficient transformation algorithm for our method to be a viable system of model-based classification (running on a single PC, the initial transformations took almost three weeks to complete). To that effect, we have implemented several optimized versions of the transformation algorithms in order to speed up the running times. A discussion of the timing results occurs in Section 5.4, but for the curious reader, they are presented in Table 1.

Upon reviewing the computation times, we surmised that much of the slow performance of our transformation algorithm could be attributed to the fact that Matlab is interpreted rather than compiled. A profiling analysis of our Matlab algorithm revealed that the vast majority of the computation time was spent running through the **for** loops listed in step 5 of Figure 1(a). We did not wish to abandon Matlab, however, since it has many benefits, such as ease of implementation and the ability to quickly test program changes. Also, for certain matrix and vector operations, we found that Matlab runs faster than a native C++ implementation.

With such considerations in mind, we set out to design a version of the transformation algorithm that removed the performance bottlenecks described above but still allowed users a seamless interface with Matlab. We created an implementation of our algorithm that was integrated into a distributed computing platform called NetSolve [2], developed at the University of Tennessee. NetSolve was designed to allow researchers who primarily use scientific computing environments like Mathematica and Matlab to harness the benefits of distributed computing. In the NetSolve framework, functions written in C or FORTRAN are integrated into NetSolve computation servers. A user can then make a call from Matlab to the NetSolve agent, which forwards the request to the appropriate server. The results are returned upon completion. The framework allows for blocking and non-blocking function calls and it is possible to implement a function on a number of servers.

The NetSolve server software can be installed on any type of machine. It need not be executed on a dedicated, high-end server, which is not likely to be available in a clinical or hospital setting. The software can be installed on a workstation and set to run in the background as a screensaver-type application when spare computing cycles are available. To improve performance and harness the potential of distributed computing, we wrote a C version of step 5 and integrated it into the NetSolve framework, which allowed us to send a request for computation to a NetSolve server, off-loading much of the work and drastically reducing the overall running times.

Finally, our MAM-classification strategy can take advantage of both parallelly-computed transformations and parallelly-built decision trees, which fits quite well into the NetSolve framework.

5 Experiments and Results

In this section we provide a description of our dataset and a summary of our experiments and results. Unless otherwise noted, all classification results are listed as the percentage of correct classification for the dataset.

5.1 Fidelity of Spatial Transformations

The first experiment was to use the algorithm given in Figure 1 to convert the original patient records into their equivalent Zernike (Z) and pseudo-Zernike (PZ) polynomial representations. The code was written in Matlab[3] and executed on an Athlon XP 1800 with 512 MB of RAM running Windows XP and Matlab 6.5R13. We computed fourth through tenth order Zernike and pseudo-Zernike transformations on four different radii values: 2.0, 2.5, 3.0 and 3.5, yielding a total of fifty-six transformations for each record.

[2] www.cs.utk.edu/netsolve/

[3] http://www.mathworks.com/products/matlab/

In addition to the transformations, we computed the residual modeling error as a function of the polynomial order for each diameter. The modeling error was calculated by determining the RMS error between the transformation and the original patient data (steps 7-9 in Figure 1(a)).

In Table 1, we provide the average transformation error of each class for 4th, 7th and 10th order transformations. We found that residual error was marginally better for a pseudo-Zernike fit when compared with a similar order Zernike polynomial fit. A more sizable difference was found between the different patient groups, however. Keratoconus was associated with residual errors greater than normal eyes and LASIK eyes had less residual error than the normal eyes. These results have anatomical validity, as a patient suffering from keratoconus is likely to have much greater distortions of corneal shape than either normal or LASIK eyes and these distortions will be more difficult to accurately model. The complex surface features seen in keratoconus will have a wider dynamic range than either of the other two patient groups studied. Likewise, normal eyes represent a wider range of anatomical shapes when compared to eyes that have undergone LASIK refractive surgery. The purpose of refractive surgery is to correct optical aberrations of the eye by altering the anatomical shape of the cornea. However, the treatment performed is typically uniform despite individual variations that may exist before treatment. As a result, corneal shape is more homogeneous in this group and is the easiest to model accurately.

We initially expected that the number of polynomial coefficients fit to the data would determine the accuracy of the spatial transform and thereby drive the classification results. While we did observe that residual error was reduced as the polynomial order of the transform increased (Table 1), the reduction in residual error did not seem to be directly related to the absolute number of polynomial coefficients. Although the pseudo-Zernike transformations were generally better for the same order, *the benefits in the reduction in residual error was modest when compared to the greater computational times required.* More importantly, this reduction in error did not translate to consistently better classification results.

	Ord	Rad	Residual Error Normal	Kera	LASIK
PZ	4	2	0.47	2.63	0.15
PZ	4	2.5	0.7	4.25	0.22
PZ	4	3	1.04	6.21	0.34
PZ	4	3.5	1.49	8.7	0.53
Z	4	2	0.49	2.68	0.21
Z	4	2.5	0.73	4.36	0.33
Z	4	3	1.08	6.43	0.6
Z	4	3.5	1.56	9.07	1.02
PZ	7	2	0.42	2.59	0.05
PZ	7	2.5	0.65	4.2	0.08
PZ	7	3	0.97	6.15	0.14
PZ	7	3.5	1.4	8.65	0.2
Z	7	2	0.44	2.58	0.1
Z	7	2.5	0.68	4.19	0.15
Z	7	3	1	6.13	0.22
Z	7	3.5	1.43	8.57	0.34
PZ	10	2	0.4	6.21	0.02
PZ	10	2.5	0.62	5.23	0.03
PZ	10	3	0.94	8.01	0.06
PZ	10	3.5	1.35	13.34	0.1
Z	10	2	0.42	2.6	0.05
Z	10	2.5	0.65	4.27	0.08
Z	10	3	0.97	6.24	0.12
Z	10	3.5	1.38	8.72	0.18

Table 1: Residual error (in μm) for each patient class by order (Ord) and radius (Rad) for each polynomial transformation.

5.2 Classification

In the next experiment, we compared eleven different classification methods to determine the one that performed best with respect to classification accuracy. The classification methods that we tested included an ID3-based [12] decision tree ($C4.5$), voted feature intervals (VFI) [7], Naïve Bayes (NB) [10] and a neural network (NN) [1]. We ran each classifier with boosting (Bst) [8], with bagging (Bag) [3] and without (W/O), except for the neural network, which we did not boost or bag. Finally, we used the above classification methods as underlying models for our MAM-classifier.

With the exception of the MAM-classifier, we conducted all of our classification experiments using the Weka Data Mining Toolkit v3.4[4] on a 2.4GHz Pentium 4 workstation with 512 MB of RAM running Windows XP and Sun Java2 v1.4.2.

For the ID3-based decision tree, we chose to use C4.5. The Weka version of C4.5 implements Revision 8, the last public release of C4.5 [12, 13, 21]. We used the following parameters in our experiments: binary splitting on attributes, confidence of 0.25, and a minimum number of 2 instances per node. With the VFI experiments, we set the bias parameter to 0.6. For the neural network, we used a multi-layer perceptron [1]. The perceptron was set to run with a validation set comprised of 30% of the instances. Training consisted of 200 iterations. For the bagging experiments, we set the bag size parameter to 100% and the number of iterations to 10. The boosting

[4] http://www.cs.waikato.ac.nz/ ml/weka/

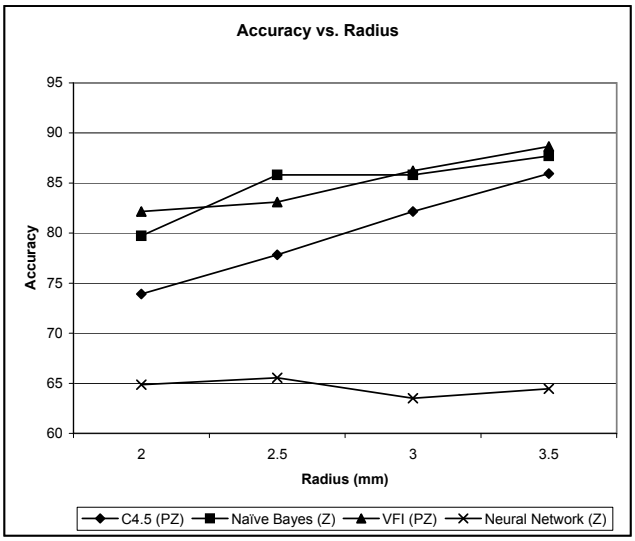

Figure 2: Accuracy vs radius for the standard classifiers tested. Graph depicts results for a 4th order transformation.

Ord	Tfrm	C4.5			VFI			NB			NN
		W/O	Bag	Bst	W/O	Bag	Bst	W/O	Bag	Bst	
4	z	88	90	91	87	86	87	88	88	88	64
5	z	88	89	89	88	87	83	86	86	86	67
6	z	86	90	91	91	89	88	84	85	84	67
7	z	86	89	91	89	89	87	79	80	80	64
8	z	84	89	90	89	88	85	76	77	76	59
9	z	84	89	91	87	86	83	76	75	76	67
10	z	87	89	90	86	85	84	73	74	73	61
4	pz	86	89	91	89	88	85	88	89	88	66
5	pz	87	89	92	90	89	88	82	83	82	67
6	pz	84	89	90	89	88	85	78	78	78	64
7	pz	88	90	91	88	87	86	73	73	73	55
8	pz	85	90	89	86	85	82	71	71	71	51
9	pz	87	89	91	85	85	84	71	70	71	49
10	pz	87	89	89	82	83	84	70	69	70	48

Table 2: Accuracy for the standard classifiers with a transformation radius of 3.5 *mm* and a testing/training split of 70/30. Ord refers to the polynomial order, Rad to the maximum radius and $Tform$ to the transformation used. Reported results averaged over ten independent runs.

experiments used the AdaBoost [8] classifier with a weight threshold of 100, again with the number of iterations set to 10. The boosted and bagged classifiers used the same parameters as the non-boosted and non-bagged versions.

In the classification experiments, we tested each method using random samples of our dataset over a number of different training/testing splits. We varied the training/testing split percentages as follows: 50/50, 60/40, 70/30, 80/20 and 90/10. All classification results are averaged over 10 different runs on different samples. Similar results were also observed with 10-fold cross-validation. We achieved the best results with our MAM-classifier when using decision trees and as such we only present the results of our MAM-classifier using C4.5, C4.5 with boosting and C4.5 with bagging.

When examining the classification results using the standard classifiers, we find that *as the radius increases, the classification accuracy increases as well.* We show these results in Figure 2. To improve the readability of the graph, we present one transformation for each classifier. We elected to use the 4th order results, but they are illustrative of the other transformations. The direct relationship between accuracy and radius is true for every classifier except the neural network, for which accuracy tends to remain constant regardless of the radius. The accuracy for the neural network was very poor in all of our experiments, generally falling between 50 and 70%. We were quite surprised by these results, as others have reported success in using neural networks on similar classification experiments [14, 15]. We are currently investigating the causes for this poor performance. However, since the neural network performed so much worse than the other methods, we will not mention it any further in our discussion. Also, since we achieved the highest classification accuracy with a radius of 3.5 *mm*, we will only refer to those results. In Table 2, we detail the results of our classification experiments on a 70/30 split of the data.

While increasing the radius tended to having a positive effect on classification accuracy for all of the classifiers, no such trend emerges when the polynomial order of the transformation is increased. For instance, the accuracy of C4.5 remains fairly stable regardless of the order, ranging from 83 to 88%. On the other hand, the accuracy of the Naïve Bayes classifier decreases drastically as the order is increased, falling from around 88% accuracy with a 4th order transformation to approximately 70% with a 10th order transformation (the same trend was observed for both the Zernike and pseudo-Zernike transformations). VFI behaves differently than both Bayes and C4.5. When classifying with VFI, accuracy is highest when using a 5th or 6th order transformation and it falls when the order deviates from those values.

The effects of boosting and bagging on classification accuracy also differ depending on the classifier. When used with C4.5, boosting and bagging improve accuracy to approximately 90%, with boosting have a greater impact (in terms of an increase in accuracy) than bagging. When boosting and bagging were used with VFI, the accuracy decreased. They had a negligi-

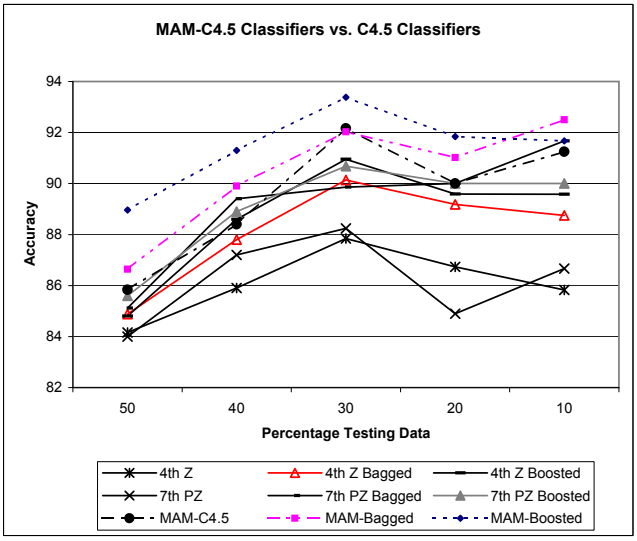

Figure 3: Comparison of the *MAM* versions of the C4.5 classifiers (bagged, boosted, without) versus a 4th and 7th order Zernike and pseudo-Zernike classifier over several training/testing splits.

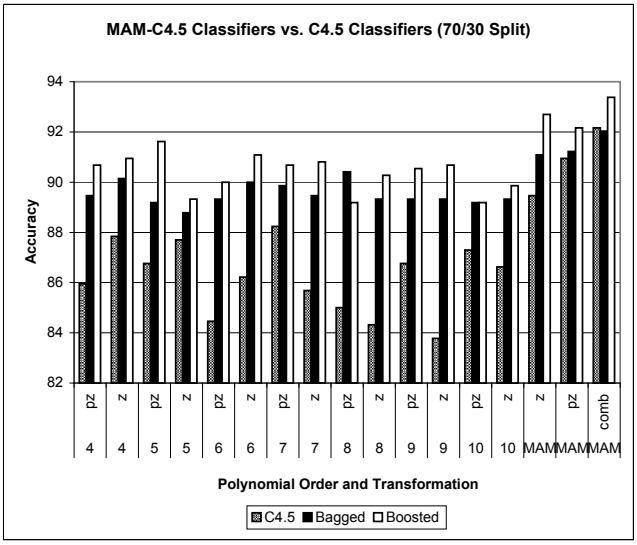

Figure 4: Accuracy for the C4.5-based MAM-classifiers as a function of polynomial order and transformation for the 70/30 training/testing split. *MAM Z* refers to a MAM-classifier where only the results of the Zernike decision trees are considered. *MAM PZ* is a similar classifier for pseudo-Zernike trees and *MAM Comb* uses the results of every tree.

ble effect on the accuracy of the Bayes classifier.

The best classification accuracy was achieved when using the MAM-classifier with the C4.5 decision trees. As shown in Figure 3, *the accuracy of the MAM-classifier was better than any of the classifiers for a single transformation order* (to improve the readability of the graph, only the results for a few of the individual trees are shown, but they are illustrative of the others). For the MAM-C4.5 classifier, the improvement in accuracy ranged from 1 to 4% over the single best individual classifier, depending on the training/testing split. Running the MAM-classification algorithm on a boosted or bagged C4.5 decision tree yielded even better results, improving accuracy 3 to 6%. We now take a closer look at the results from the 70/30 split.

Figure 4 shows a breakdown of classification accuracy as a function of each transformation and polynomial order as well as the results from the MAM-classifiers for C4.5 with boosting, with bagging, and without. For each version of C4.5 tested (boosted, bagged, without), the MAM-classifiers based on the individual transforms (Zernike, pseudo-Zernike) produced a higher accuracy than any individual order decision tree and the combined version of the MAM-classifier produced the best results.

After running our initial classification experiments, we repeated our procedures using random forests [4]. The random forest algorithm constructs a decision tree by using random samples of the data and random samples of the features at each node. This method is similar to our MAM-classification strategy, though we use subsets of the transformation models, not individual features. We found that on average, random forests performed worse than our MAM classification strategy and about as well as regular C4.5. We also found that it takes longer to build a random forest and that the classifier suffers from a lack of interpretability. Our users rate the interpretability of a random forest classifier to be about the same as a neural network. For these reasons, we chose not to include any further discussion into random forests.

5.3 Visualization

Using the algorithm described in Section 4.3, we took one of the C4.5 decision trees and constructed a decision surface for the distinction between non-diseased corneas and those having keratoconus (we removed the LASIK corneas before creating the decision tree). The surface, presented in Figure 5, was constructed by taking the splitting attributes of the decision tree that led to a classification of keratoconus and compiling a vector of these coefficient values. If an attribute was important in the discrimination of keratoconus, it was assigned a value equal to the splitting criteria from the decision tree. Attributes that did not contribute to the classification decision were assigned a value of 0. This surface was designed to distinguish between two different classes. For a multi-class dataset,

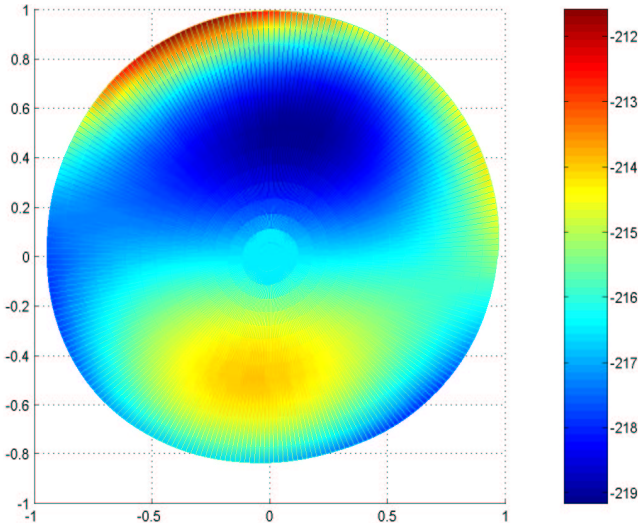

Figure 5: Surface constructed by combining decision tree attributes and values that contributed to keratoconus classification. Scale represents surface elevation in microns. Red-shifted colors represent greater elevation; blue-shifted colors represent lower.

it would be necessary to construct multiple surfaces to differentiate between all of the classes.

This surface is a false color elevation map similar to weather maps that show (x,y) spatial coordinates and use blue to red colors to represent higher or lower atmospheric pressure. Here we use color to show elevation of the corneal surface in microns relative to the corneal apex over a 7 *mm* diameter region (normalized for a unit circle).

To a domain expert this result has face validity. *We find that the same features normally associated with this degenerative condition were important for decision tree classification*–increased surface elevation localized in the lower quadrant, for instance.

There are several extensions to this technique that could be used to greatly assist clinical decision support. For instance, if a patient was classified as having keratoconus, it is possible to create a picture placing the topographic image of a patient's cornea next to the decision surface, allowing a clinician to visually inspect the criteria used to classify the patient as diseased. Though this particular decision surface was constructed from a C4.5 decision tree, it is possible to use our method with bagged, boosted, or MAM classifiers by creating a surface for every tree that is used by the meta-classifier.

5.4 Optimization of Transformation Algorithms

We used the following setup to test our high-performance implementations. Matlab was installed on a PC with an Athlon XP 1800 processor and 512 MB RAM running Windows XP (Same PC describe in Section 5.1). NetSolve was installed on a PC with an Athlon XP 1800 processor and 512 MB RAM running Debian Linux with a 2.4.x kernel. Both computers were on the same network connected to a 100 Mbps Ethernet switch. The NetSolve server was started where it would wait for incoming requests.

Order		4		7		10	
Time		M	NS	M	NS	M	NS
Z	2	10	3	31	2.2	69	1.8
Z	2.5	13	7	38	1.3	84	2
Z	3	15	4.7	45	1.4	101	1.8
Z	3.5	17	1	52	2	116	3.5
PZ	2	23	2.5	82	2	200	4.5
PZ	2.5	29	2.8	101	2.3	246	4.7
PZ	3	34	2.1	121	2.6	294	5.7
PZ	3.5	39	2.8	139	4.3	339	6.5

Table 3: Transformation running times (in seconds). Table shows results for 4th, 7th and 10th order transformations on each radius. M refers to the unoptimized Matlab algorithm, NS to the NetSolve version.

Rather than test performance on the entire dataset, a subset of 10 records was used. This subset contained records from each patient class. Our original Matlab experiments showed that for a given set of parameters, the running time was the same across all of the patient records. Thus, it is not a problem to use a small subset of entire dataset in order to model the performance of each implementation. As in the original experiment, each transformation was computed for 4th-10th order for the four radii values (2.0, 2.5, 3.0, 3.5). A Matlab script was written to compute the Zernike and pseudo-Zernike transformations for each radius and order over the subset of patient records. Calls to NetSolve were made in order to compute the coefficient matrix for each transformation.

Table 3 provides the running times for both the unoptimized transformation algorithm and the distributed NetSolve version. In both cases, as the radius increases, the computation times grows. This increase is a result of the larger number of data points that are taken into consideration. In the unoptimized version, there is a also a marked difference between the time needed to compute a Zernike polynomial compared to the time needed to compute a pseudo-Zernike polynomial. This time difference is due to the increased number of coefficients that must be computed for a pseudo-Zernike transformation compared to a Zernike transformation for a given order. For instance, a 10th order Zernike polynomial consists of 66 coefficients, while a 10th order pseudo-Zernike polynomial is composed of 121.

When examining the data, it becomes quite clear that the rate-limiting step in the original Matlab algorithm was the two **for** loops where the transformation coefficients were calculated (step 5, Figure 1(a)). The Matlab implementation with NetSolve achieved a speed up of anywhere from 2x to 50x over the original Matlab implementation. The speed up of 2x was for a 4th order Zernike transformation at a radius of 2.0. For a 10th order pseudo-Zernike transformation at a radius of 3.5, the difference in computation time is staggering. The unoptimized version took almost six minutes to complete. The version integrated with NetSolve took roughly six *seconds*. Since the speed up factor tends to increase as the polynomial order of the transformation increases, the computational benefits would likely be even greater for transformations involving polynomial orders larger than ten.

To show the benefits of a distributed implementation, we calculate the time required to serially compute all of the spatial transformations for the MAM-classifier (using times from Table 3): 42 seconds. In a completely distributed setting, the time required is equal to the time of the slowest transformation. The rate-limiting step in this case is computation of a 10th order pseudo-Zernike transformation, which takes 6.5 seconds. This yields a 6x speed up over the serial version. Similarly with classification, we create decision trees for each record simultaneously, yielding a linear speed up (classification time is roughly the same across each order, approximately 1 second). *Thus, using NetSolve, we can classify a single record using our MAM classification strategy in roughly 7 seconds, compared with almost a minute in a serial environment.*

6 Conclusions

In this work we have provided a number of extensions to a previously conducted study in the classification of an eye disease affecting the cornea. The first extension involved an examination of the use of Zernike and pseudo-Zernike polynomials as a method of spatially transforming our data into a form that could easily be used as input for a classifier. We tested each transformation over a number of polynomial orders and studied the trade-offs between classification accuracy, computation time and transformation fidelity. Our general finding was that unlike results reported in the literature [9], *there was no special benefit to using pseudo-Zernike polynomials over Zernike polynomials.* Classification accuracy using pseudo-Zernike polynomials was roughly the same or slightly worse than a Zernike polynomial of the same order. In the face of the additional time needed to compute a pseudo-Zernike transformation, we cannot recommend them as a transformation method.

	Interpretability	Accuracy	Speed
Neural Network	—	—	—
C4.5	++	+	++
Naïve Bayes	+	±	++
VFI	+	+	++
MAM-C	+	++	++

Table 4: Summary of comparisons

The second contribution of this work involves the creation of a novel meta-classification strategy that takes advantage of multiple classification models to increase the accuracy of an individual classifier. We compared our strategy and a number of standard classifiers and meta-classification strategies and found that a MAM-classifier provides superior results on our data. We would like to examine the general purpose benefits in a future study.

We found that with the traditional methods, C4.5-based decision trees provided the highest, most consistent classification results, especially when coupled with meta-learning strategies such as boosting or bagging. These results generally agree with the results of our previous work. We found that other classification methods such as VFI or Naïve Bayes sometimes performed as well as C4.5-based strategies, but the classification results for those methods were less stable across the different polynomial orders that we tested. Also, when boosted or bagged, the accuracy of the VFI and Bayes classifiers tended to remain the same or even decrease. *The highest classification results, however, were obtained when combining our MAM-classification strategy with the C4.5-based classifiers.* We were able to achieve an increase in accuracy of up to 6%, leading us to believe that the MAM-classification strategy can be very useful on any problem where classification input involves features derived from a model-based transformation.

In Table 4 we provide a summary comparison of the classifiers tested in this study. In general, we found the C4.5 classifier to be the most interpretable, while our MAM-C classifier had the highest accuracy. Both VFI and Naïve Bayes were accurate on certain orders, but they were not very stable, with Bayes less stable than VFI. The accuracy of the neural network was very poor, no matter what the order, and there is no easy way to interpret the underlying classification model. The time required by the C4.5, Bayes, and VFI classifiers was minimal, especially when compared to the running time of the neural network. Our MAM-C classifier was marginally slower than the tree-based classifiers, but only because it had to wait for the underlying classifiers

to finish before proceeding.

We would like to conduct further experiments with our MAM-classification strategy to determine whether using subsets of the decision trees could provide better results. Using subsets might also allow us to use it with non-C4.5-based classifiers. The accuracy of the simple Naïve Bayes classifier fell drastically as the polynomial order of the transformation was increased, meaning a MAM-classifier that uses trees from every order will have a lower accuracy than an individual low-order tree. Combining just the 4th or 5th order trees might provide the increase in accuracy that we were able to achieve with the MAM-C4.5 classifier. The same principle applies to using the 5th and 6th order trees with VFI.

The third contribution of our work is the development of a *novel method of visualizing our decision tree results.* The motivation behind this extension was to design a tool that can be used by clinicians to aid in patient diagnosis and decision support. By taking advantage of the unique properties of Zernike polynomials, we are able to create a decision surface which reflects the geometric features that discriminate between diseased and non-diseased eyes. If a patient was diagnosed as diseased, a clinician could take the topographic image of a patient's eye and our decision surface and visually determine the features which led to the diagnosis. In Table 4 we give a higher interpretability rating to classifiers that can be easily converted to a decision surface. With the meta classifiers, including our MAM-classifier, multiple surfaces would need to be created, which slightly decreases the ease of interpretability.

The final extension that we show involves a high-performance implementation of our transformation that results in a drastic reduction in the overall running time of our algorithm and also provides a foundation for a distributed implementation of our classification system within a hospital or health center.

As a final note, we do not intend to limit our experiments to the classification of keratoconus. We intend to expand this work to provide a generalized approach that can be leveled against other diseases that affect the visual field, diseases such as glaucoma or the onset of cataracts. Also, while we have presented an in-depth study of classifying clinical eye disease data, this work has implications that can be applied in a broad sense toward any problem that deals with the model-based mining of data.

7 Acknowledgments

NIH-NEI T32-EY13359 and American Optometric Foundation William Ezell Fellowship, Ocular Sciences Inc. (MT); NSF Career Grant IIS-0347662(SP); NIH-NEI R01-EY12952 (MB)

References

[1] C. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, Oxford, 1995.

[2] M. Born and E. Wolf. *Principles of optics : electromagnetic theory of propagation, interference and diffraction of light.* Pergamon Press, NY, 6th ed, 1980.

[3] L Breiman. Bagging predictors. *Machine Learning*, 26(2):123–140, 1996.

[4] L Breiman. Random Forests. *Machine Learning*, 45 (1):5-32, 2001.

[5] C-W Chong, P Raveendran, and R Mukundan. An efficient algorithm for fast computation of pseudo-zernike moments. *Intl Journal of Pat. Recog. and AI*, 17(6):1011–1023, 2003.

[6] I Daubechies. *Ten Lectures on Wavelets.* SIAM, 1992.

[7] G Demiroz and HA Guvenir. Classification by voting feature intervals. In *Euro Conf on ML*, 1997.

[8] Y Freund and R Schapire. Experiments with a new boosting algorithm. In *13th Intl Conf on ML*, 1996.

[9] D Iskander, M Morelande *et al* Modeling of corneal surfaces with radial polynomials. *IEEE Trans Biomed Eng*, 49(4):320–328, 2002.

[10] GH John and P Langley. Estimating continuous distributions in (Bayesian) classifiers. In *11th Conf on Uncertainty in AI*, pages 338–345, 1995.

[11] AF Laine. Wavelets in temporal and spatial processing of biomedical images. *Annu. Rev. Biomed. Eng.*, 02:511–550, 2000.

[12] JR Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

[13] JR Quinlan. *C4.5 : programs for machine learning.* Morgan Kaufmann Publishers, San Mateo, Calif., 1993.

[14] MK Smolek and SD Klyce. Current keratoconus detection methods compared with a neural network approach. *Invest Ophthalmol Vis Sci*, 38(11):2290–9., 1997.

[15] MK Smolek and SD Klyce. Screening of prior refractive surgery by a wavelet-based neural network. *J Cataract Refract Surg*, 27(12):1926–31., 2001.

[16] C-H Teh and R Chin. On image analysis by the methods of moments. *IEEE Trans. on Pat Analysis and Mach Intel*, 10(4):496–513, 1988.

[17] L Thibos, R Applegate *et al* Standards for reporting the optical aberrations of eyes. In *TOPS*, Santa Fe, NM, 1999. OSA.

[18] M Twa, S Parthasarathy *et al* Automated classification of keratoconus: A case study in analyzing clinical data. In *SIAM Intl. Conf on DM*, 2003.

[19] vF Zernike. Beugungstheorie des schneidenverfahrens und seiner verbesserten form, der phasenkontrastmethode. *Physica*, 1:689–704, 1934.

[20] SE Wilson and SD Klyce. Screening for corneal topographic abnormalities before refractive surgery. *Ophthalmology*, 101(1):147–52., 1994.

[21] IH Witten and E Frank. *Data mining : practical machine learning tools and techniques with Java implementations.* Morgan Kaufmann, 2000.